Informe Desafío II

Juan Manuel Álvarez Cajiao

Informática 2

Aníbal José Guerra Soler

Augusto Enrique Salazar Jiménez

Universidad de Antioquia

Facultad de Ingeniería

Medellín

2024

Análisis del problema

El desafío del proyecto es simular el funcionamiento de una red nacional de estaciones de servicio de la empresa ficticia "TerMax", la cual gestiona el suministro de combustibles en varias ubicaciones de Colombia. Cada estación de servicio está equipada con un tanque central para almacenamiento de combustibles y varios surtidores que manejan las ventas.

**Requerimientos principales:**

* El sistema debe gestionar una red de estaciones de servicio.
* Cada estación tiene un único tanque y varios surtidores.
* Se deben registrar y gestionar las ventas de tres tipos de combustibles: Regular, Premium y EcoExtra.
* El sistema debe ser capaz de simular ventas, actualizar los niveles de los tanques, manejar el estado de los surtidores (activación/desactivación), y calcular las ventas por tipo de combustible.
* Además, se requiere verificar posibles fugas en los tanques.

**Consideraciones para la solución:** La solución propuesta se basó en una arquitectura orientada a objetos, donde se modelan las entidades principales: *RedNacional, EstacionDeServicio, Tanque, Surtidor y Venta.* Estas clases representan los componentes esenciales para simular la operativa de las estaciones de servicio. Las decisiones claves del diseño incluyeron:

* **Gestión modular**: Cada componente es independiente, permitiendo agregar o eliminar estaciones y surtidores sin afectar al resto del sistema.
* **Simulación de ventas**: El sistema permite registrar ventas en tiempo real, calculando el consumo de combustible en el tanque y manteniendo un historial de transacciones.
* **Flexibilidad**: La red se puede expandir dinámicamente, ajustándose a la necesidad de agregar o eliminar estaciones y surtidores.

Diagrama de clases de la solución planteada

El diagrama de clases se organizó de la siguiente manera:

* **RedNacional**: Representa la red de estaciones de servicio. Gestiona las operaciones globales como agregar o eliminar estaciones, y las ventas en general.
* **EstacionDeServicio**: Cada estación tiene un tanque de combustible y múltiples surtidores. Lleva a cabo operaciones específicas como agregar o eliminar surtidores, simular ventas y gestionar el tanque.
* **Tanque**: Cada estación tiene un único tanque que almacena combustible y puede disminuir su nivel en función de las ventas.
* **Surtidor**: Un surtidor registra las ventas de combustible, interactúa con el tanque para actualizar la cantidad disponible y mantiene un historial de transacciones.
* **Venta**: Representa una transacción individual, registrando la cantidad de litros vendidos y el tipo de combustible.

[*DIAGRAMA*](file:///C:\Users\Dell\OneDrive\Desktop\diagrama_de_clases_DESAFIO2.drawio.pdf)

Descripción de la lógica de tareas

A continuación, se detallan algunas de las tareas no triviales:

* **Simulación de ventas**: El sistema simula ventas reales al ingresar una cantidad de litros y el tipo de combustible. El surtidor elegido verifica si el tanque tiene suficiente combustible, reduce la cantidad de combustible y registra la venta en su historial.

**Lógica:**

* 1. El surtidor recibe la solicitud de venta.
  2. Consulta al tanque si tiene suficiente combustible.
  3. Si es posible, reduce el nivel del tanque y registra la venta.
  4. Si no es posible, muestra un error al usuario.
* **Gestión dinámica de surtidores**: El sistema permite agregar nuevos surtidores a una estación de servicio o eliminarlos si están inactivos.

**Lógica:**

* 1. Se ingresa el código de la estación de servicio.
  2. Se agrega un surtidor nuevo si hay capacidad, o se elimina un surtidor inactivo.
  3. Si se elimina, los surtidores posteriores se ajustan para llenar el espacio vacío.
* **Detección de fugas**: Cada estación realiza chequeos periódicos del nivel de combustible en el tanque y verifica si hay una discrepancia en los niveles registrados versus las ventas reportadas.

**Lógica:**

* 1. Comparar el nivel actual del tanque con el registro histórico de ventas.
  2. Si el nivel esperado es mayor al nivel real (con un margen permitido), se reporta una posible fuga.

Algoritmos implementados

Ejemplo del algoritmo de eliminación de surtidor:

void EstacionDeServicio::eliminarSurtidor(int codigoSurtidor) {

int indiceSurtidor = -1;

for (int i = 0; i < numSurtidores; ++i) {

if (surtidores[i]->getCodigo() == codigoSurtidor) {

indiceSurtidor = i;

break;

}

}

if (indiceSurtidor == -1) {

cout << "Error: Surtidor no encontrado." << endl;

return;

}

if (surtidores[indiceSurtidor]->isActivo()) {

cout << "Error: No se puede eliminar un surtidor activo." << endl;

return;

}

delete surtidores[indiceSurtidor];

// Eliminar el surtidor moviendo los elementos

for (int i = indiceSurtidor; i < numSurtidores - 1; ++i) {

surtidores[i] = surtidores[i + 1];

}

numSurtidores--;

cout << "Surtidor eliminado correctamente" << endl;

}

Este algoritmo incluye validaciones para asegurar que el surtidor solo se elimine si está inactivo y si el código proporcionado corresponde a un surtidor existente. Tras la eliminación, se reorganiza el arreglo de surtidores.

Problemas de desarrollo afrontados

Durante el desarrollo del sistema se presentaron varios problemas, entre ellos:

1. **Manejo de memoria dinámica**: Uno de los desafíos fue implementar correctamente los arreglos dinámicos para las estaciones de servicio y surtidores, asegurando que se redimensionaran de forma eficiente y sin pérdida de datos.
2. **Interacción entre clases**: Debido a la interdependencia de las clases EstacionDeServicio, Tanque, y Surtidor, fue necesario implementar correctamente la comunicación entre ellas para simular las ventas sin errores. Esto implicó revisar la lógica de acceso a los datos del tanque y surtidores.
3. **Verificación de estado**: Al permitir que los surtidores fueran activados o desactivados, fue importante manejar los estados de cada surtidor y permitir solo operaciones válidas.

Evolución de la solución y consideraciones

A medida que el desarrollo del sistema avanzaba, la solución fue evolucionando en términos de modularidad y manejo de memoria. Inicialmente, la gestión de las estaciones y surtidores se realizó mediante arreglos de tamaño fijo, pero se decidió utilizar arreglos dinámicos para permitir mayor flexibilidad y escalabilidad.

Como proyecto prácticamente finalizado, se hizo uso de manejo de arreglos dinámicos de punteros a objetos, con el objetivo de manipularlos de una manera correcta que no generara errores al combinar archivos de heap con archivos de stack.